УДК 004.89

М.А. Князева, В.А. Тимченко
Институт автоматики и процессов управления ДВО РАН, г. Владивосток, Россия
mak@imcs.dvgu.ru, rakot2k@mail.ru.

Преобразование графовых структур представления информации

В работе изложен подход к преобразованию информации, представленной графовыми структурами. Представлена концептуальная схема преобразования информации на основе проекций графовых структур. Приведены модели, в соответствии с которыми описываются графовые структуры и проекции, которые являются спецификацией на преобразование графов. Предложенный подход реализован в прототипе, предназначенном для перевода программ с одного процедурного языка программирования на другой. Прототип поддерживает языки Pascal, C, язык моделей структурных программ и разработан в рамках системы преобразований программ. Прототип реализован в среде программирования Java.

Введение

На сегодняшний день компьютерная обработка информации является одним из критических видов деятельности в большинстве прикладных и теоретических областей. Этот вид деятельности включает задачи по получению, инженерии, хранению, управлению и использованию различных видов данных и знаний.

При этом происходит постоянный обмен информацией между компьютерными системами, их компонентами, людьми, организациями. Видами информации могут быть искусственные языки (языки программирования, языки спецификаций, языки описания структур данных и т.д.), знания, данные, программы, представленные на этих языках. При передаче информации могут изменяться такие ее аспекты, как формат представления (изменение структуры, модели, языка представления), интерпретация, уровень детализации.

Примером может служить преобразование информации на разных этапах разработки информационных систем. Концептуальные модели данных, независимые от реализации, отображаются на релевантные им модели данных, ориентированные на реализацию в конкретной операционной среде. Например, семантическая модель данных типа «сущность-связь» отображается на реляционную модель данных, представление на языке UML − в представление на языке XML [1].

Другим ярким примером может служить по-прежнему остро востребованная задача трансляции компьютерных программ с одного языка представления этих программ на другой. Она возникает в таких областях, как:

– разработка систем, предназначенных для анализа, оптимизации и распараллеливания программ, в особенности таких, которые работают с несколькими языками исходных текстов программ. При этом выполняется трансляция анализируемой программы на исходном языке во внутреннее представление, используемое затем для анализа и преобразований [2], [3];

– программный реинжиниринг, который занимается в частности проблемой перевода программ с устаревших языков на новые;

– разработка языков программирования, ориентированных на решение задач в конкретной предметной области (domain-specific languages, DSLs). Это, как правило, компактные языки программирования, предназначенные для решения специфических для некоторой предметной области задач в противовес языкам программирования общего назначения, цель создания которых – решение вычислительных задач в любой предметной области [4]. Они получают все большее распространение в связи с тем, что число современных прикладных задач стремительно растет.

Задача построения трансляторов всегда определяется как высокотехнологичная и времяемкая, которая может быть решена только соответствующими специалистами. Это препятствует массовой разработке средств трансляции.

Многие задачи, связанные с преобразованием разных видов информации, еще находятся на стадии исследования, различные исследовательские группы разрабатывают методы их решения разной эффективности. Значительный вклад в исследование этой проблемы внесли В.Ш. Кауфман, A. Königs, H.-J. Kreowski, S. Kuske, A. Schürr и другие.

Ввиду сложности этих задач, они в основном рассматриваются как самостоятельные, методы их решения разрабатываются независимо друг от друга. На стадии исследований с целью проверки этих методов создаются макеты программных систем. Поскольку при создании макетов проблема их совместимости не рассматривается, разработчики, концентрируя свое внимание на методах решения, часто выбирают специфическое представление используемой информации. Получаемые в результате компьютерные системы часто оказываются несовместимыми между собой, замыкаются на своих предметных областях, оказываются не способными взаимодействовать друг с другом для решения задач преобразования информации, возникающих на стыке предметных областей. Сопровождение зачастую оказывается совершенно нерентабельным по срокам и трудозатратам. В результате они не могут быть использованы для решения задач другой предметной области и задач на стыке предметных областей, вследствие чего для этого приходится разрабатывать новое программное обеспечение.

Другой проблемой разрабатываемых таким образом макетов программных систем является то, что большинство из них, как правило, не доводятся до уровня распределенных сетевых приложений (функционирующих в среде Интранет/Интернет), а так и остаются на уровне настольных персональных версий и используются только там, где разрабатываются. Это резко сокращает доступность, масштаб практического применения и, как следствие, востребуемость таких средств, что в свою очередь препятствует накоплению опыта их практического использования и тормозит темпы их дальнейшего развития.

Любая информация может быть представлена в виде графовых структур (например, представление информации в виде семантических сетей), а потому преобразование информации может рассматриваться как преобразование графовых структур.

Графовые структуры данных являются естественным и наглядным средством представления сложных структур и процессов. Это позволяет широко использовать их в компьютерных системах при решении различных задач. Во многих из этих задач графы используются для представления данных неоднородной структуры. К таким задачам относятся представление деревьев абстрактного синтаксиса программ в трансляторах языков программирования, описание структуры и обработки объектных моделей документов, обработка сложных структур данных предметной области в прикладных задачах и т.п.

Большинство моделей, над которыми проводятся преобразования, представляются в виде графовых структур. Преобразования описываются в терминах графов и операций над ними и основываются на правилах, в результате выполнения которых порождается новый граф на основе заданного исходного.

Данная работа посвящена описанию концептуальной схемы преобразования информации, представленной графовыми структурами, а также описанию модели преобразования графовых структур и результатов экспериментального исследования прототипа программного средства при решении практических задач.

Работа выполнена при финансовой поддержке ДВО РАН в рамках Программы № 2 Президиума РАН «Интеллектуальные информационные технологии, математическое моделирование, системный анализ и автоматизация», проект 09-I-П2-04 «Развитие систем управления базами знаний с коллективным доступом».

Концепция преобразования информации на основе проекции графовых структур

Рассмотрим концептуальную схему преобразования информации, представленной графовыми структурами, в которой на основе видов используемой информации выделяется три уровня (рис. 1). Третий (самый верхний) уровень объединяет модели, в соответствии с которыми формируется информация второго уровня. Второй уровень объединяет описание графовых структур и описание структурных проекций. Описание графовых структур является метаинформацией по отношению к информации первого уровня, объединяющего, в свою очередь, информацию, которая подвергается собственно преобразованию.

Основная идея подхода состоит в том, чтобы представить исходную и целевую информацию в виде графовой структуры. На рис. 1 им соответствуют компоненты: *Описание графовой структуры исходной информации*, *Описание графовой структуры целевой информации*. Это представление включает в себя:

– определение множества понятий описываемой предметной области (вершины графа). Каждое понятие, за исключением понятий, соответствующих терминальным вершинам графа, определяется через множество некоторых других понятий, при этом подразумевается, что понятия, через которые определено некоторое другое понятие, входят в содержание последнего;

– во множестве понятий существует некоторое выделенное понятие, называемое аксиомой, в которое не входит ни одна дуга. В одну и ту же вершину может входить несколько дуг;

– определение множества направленных бинарных отношений (им соответствуют дуги графа), в которых понятия между собой состоят. Понятия связываются между собой направленными отношениями по принципу: понятие, из которого выходит дуга, определено через понятие, в которое дуга входит;

– дополнительная информация о понятии или отношении может быть посредством множества значений атрибутов, связанных с этим понятием или отношением. Множество связанных атрибутов может быть пустым.

При работе с информацией бывает удобно иметь возможность преобразовывать ее к виду, понятному человеку, причем не в виде сети вершин, а в виде привычного текстового описания. Наряду с этой ситуацией имеет место обратная ситуация, когда по текстовому представлению информации необходимо получить ее представление в виде графовой структуры. Примером этого является преобразование текста программы в ее дерево абстрактного синтаксиса. Если необходимо занести в базу графовых структур имеющуюся информацию и существует уже описание графовой структуры

этой информации, то проще построить сеть понятий, соответствующую заносимой информации, автоматически, чем вручную, с помощью средства редактирования порождать эту сеть под управлением описания графовой структуры этой информации. Таким образом, необходимо иметь средства для решения этих взаимно противоположных задач – задачи синтеза и задачи анализа текстов.

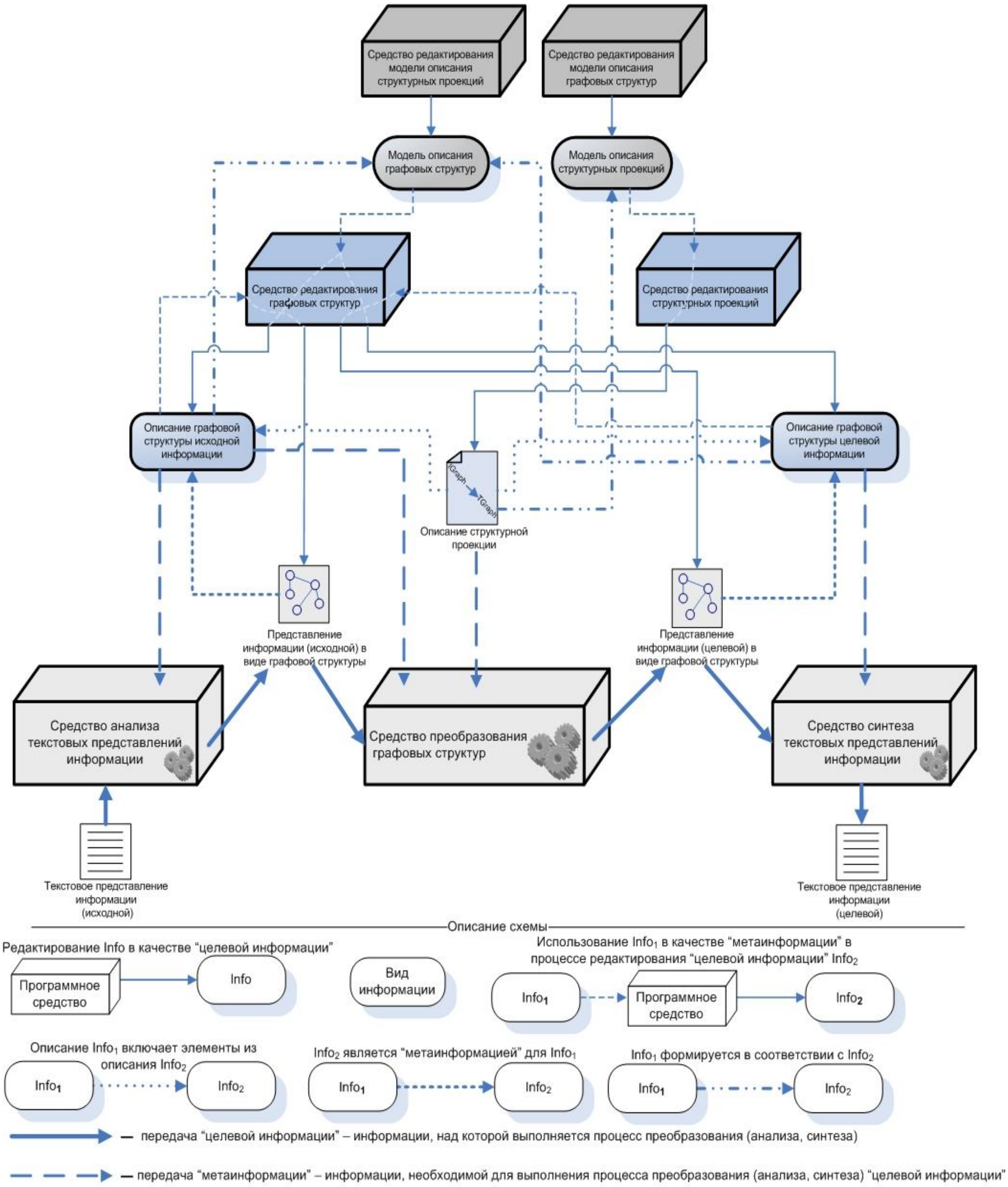


Рисунок 1 – Концептуальная схема преобразования информации, представленной графовыми структурами

Чтобы иметь возможность работать с текстовым представлением информации, необходимо описать связь графовой структуры с элементами ее конкретного синтаксиса.

Эта связь определяет содержание правильных конструкций языка текстового представления этого графа с точки зрения синтаксиса этого языка. Связь графовой структуры с ее конкретным синтаксисом содержит такие элементы языка, как пунктуационное наполнение, порядок следования слов и т.д.

Данный вид информации ограничивает способ выражения в виде текста того смысла, который заложен в описании графовой структуры, поэтому далее эта связь будет называться синтаксическими ограничениями.

Графовые структуры описываются в соответствии с *Моделью описания графовых структур*. Данная модель в частности регламентирует правила задания синтаксических ограничений и используется *Средством редактирования графовых структур* для описания графовых структур в терминах этой модели. В свою очередь, *Модель описания графовых структур* формируется и редактируется с помощью *Средства редактирования модели описания графовых структур*.

Описание графовой структуры информации и описание ее синтаксических ограничений необходимы для осуществления процедуры синтаксического анализа текстового представления информации на заданном языке и формирования представления этой информации в виде графовой структуры, а также обратной процедуры – синтеза текстового представления информации по ее представлению в виде графовой структуры. Эти процедуры выполняются соответственно *Средством анализа текстовых представлений информации* и *Средством синтеза текстовых представлений информации*.

Представление информации в виде графовой структуры (Целевой граф – Гц) является «*целевой*» (это информация более низкого уровня общности, называемая «*целевой информацией*») графовой структурой по отношению к *Описанию графовой структуры* (Метаграф – Гм), представляющей собой информацию высокого уровня общности, называемую «*метаинформацией*». Это означает, что вершины Гц, представляющие понятия, входящие в объемы понятий, представленных вершинами из Гм, являются экземплярами этих вершин из графовой структуры Гм, тогда как сами вершины графовой структуры Гм являются прототипами для вершин из графовой структуры Гц. Например, граф Гм содержит вершину «имя», а граф Гц содержит вершины «Иванов», «Александров». Эти последние вершины представляют понятия, входящие в объемы понятия, представленного вершиной «имя», из Гц и являются экземплярами этой вершины. Таким образом, можно говорить, что Гц описан в терминах Гм, или (что то же самое), что «метаинформация» представляет язык для описания «целевой информации».

Например, пусть задан фрагмент описания графовой структуры, представляющий данные о пациенте (рис. 2).

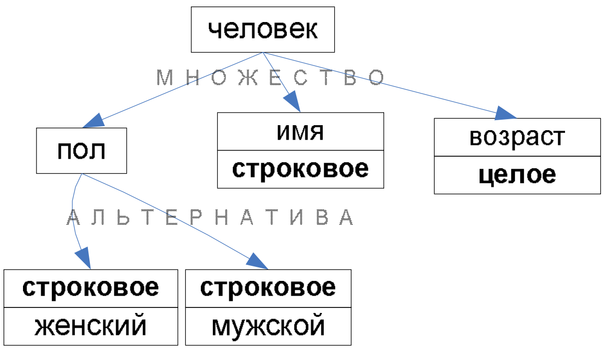


Рисунок 2 – Фрагмент графовой структуры, задающий описание пациента

Описание конкретного пациента «Пациент 1» будет выглядеть так, как показано на рис. 3.

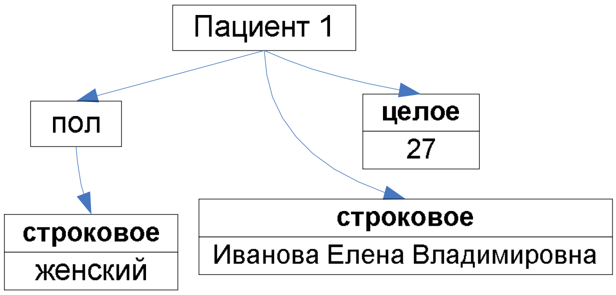


Рисунок 3 – Фрагмент графовой структуры, описывающий «Пациента 1»

Задавая правила отображения этой информации в конкретную текстовую форму (синтаксические ограничения графовой структуры, задающей описание пациента), можно получить, например, представление о «Пациенте 1» в следующем виде:

Иванова Елена Владимировна, жен., 27 лет.

Семантика преобразования задается *Описанием структурной проекции* графовой структуры исходного представления информации на графовую структуру целевого представления. Описание проекции фиксирует множество соответствий между фрагментами исходной и целевой графовых структур. Для описания проекций разработан язык описания структурных проекций.

Структурные проекции описываются в соответствии с *Моделью описания структурных проекций*. Данная модель используется *Средством редактирования структурных проекций* для описания проекций в терминах этой модели. В свою очередь, *Модель описания структурных проекций* формируется и редактируется с помощью *Средства редактирования модели описания структурных проекций*.

Преобразование информации, представленной графовыми структурами, на основе заданной проекции *Описания графа исходной информации* на *Описание графа целевой информации* выполняет *Средство преобразования графовых структур*.

Модель преобразования графовых структур на основе описания проекции

Модель преобразования графовых структур на основе описания проекций *GTM* представляет собой пару (*GDM, MDM*), где *GDM* – модель описания графовых структур, *MDM* – модель описания структурных проекций.

Модель описания графовых структур *GDM* = (*Concepts, Relations, Attributes, Axiom_Concept, Id_Concept, Const_Concept, Syntax_Restrictions*)

$Concepts = \{Concept_i\}_{i=1}^{conceptscount}$ – конечное непустое множество понятий. Каждое понятие *Concept* описывается своим уникальным именем. Имя понятия представляет собой непустую последовательность символов и идентифицирует определенный класс объектов описываемой предметной области.

$Relations = \{Relation_i\}_{i=0}^{relationscount}$ – конечное, возможно пустое, множество отношений. Каждое отношение $Relation_i$ является направленным бинарным отношением (дугой), связывающим два понятия, и описывается следующим образом: $Relation_i$ = (*Relation_Name*, *Begin_Concept*, *End_Concept*).

Relation_Name – имя отношения, которое представляет собой непустую последовательность символов.

Begin_Concept – имя понятия, из которого дуга исходит – понятие-начало отношения. $Begin_Concept \in Concepts$.

End_Concept – имя понятия, в которое дуга входит – понятие-конец отношения. *End_Concept* ∈ *Concepts*.

$Attributes = \{Attribute_i\}_{i=0}^{attributescount}$ – конечное, возможно пустое, множество атрибутов. Каждый атрибут $Attribute_i$ представляет некоторое свойство понятия и описывается следующим образом: $Attribute_i$ = (*Attribute_Name*, *Attribute_Argument*, *Attribute_Value*).

Attribute_Name – имя атрибута, представляющее собой непустую последовательность символов.

Attribute_Argument = (*Attribute_Argument_Type*, *Attribute_Argument_Value*) – аргумент атрибута описывается своим типом *Attribute_Argument_Type* и значением *Attribute_Argument_Value*.

Attribute_Argument_Type = {"Понятие", "Отношение", "Идентификатор", "Константа"}.

Attribute_Argument_Value ∈ *Concepts* ∪ *Relations* ∪ *Identifiers* ∪ *Constants*.

Identifiers – конечное, возможно пустое, множество всех идентификаторов, присутствующих в исходном представлении информации.

Constants – конечное, возможно пустое, множество всех констант и константных значений, присутствующих в представлении информации.

Аргументом атрибута может быть понятие (*Attribute_Argument_Type* = "Понятие"), отношение (*Attribute_Argument_Type* = "Отношение"), идентификатор (*Attribute_Argument_Type* = "Идентификатор") или константа (*Attribute_Argument_Type* = "Константа").

Attribute_Value = (*Attribute_Value_Type*, *Attribute_Value_Value*) – значение атрибута описывается своим типом *Attribute_Value_Type* и значением *Attribute_Value_Value*.

Attribute_Value_Type = {"Понятие", "Идентификатор", "Константа", "Строковое", "Целое", "Вещественное", "Логическое"}.

Attribute_Value_Value ∈ *Concepts* ∪ *Relations* ∪ *Identifiers* ∪ *Constants* ∪ *String* ∪ *Integer* ∪ *Real* ∪ *Boolean*.

String – множество строк.

Integer – множество целых чисел.

Real – множество вещественных чисел.

Boolean – множество {Истина, Ложь}.

Значением атрибута может быть понятие (*Attribute_Value_Type* = "Понятие"), отношение (*Attribute_Argument_Type* = "Отношение"), идентификатор (*Attribute_Value_ Type* = "Идентификатор"), константа (*Attribute_Value_Type* = "Константа"), последовательность символов, интерпретируемых как строковая константа (*Attribute_Value_Type* = "Строковое"), целое число из множества целых чисел (*Attribute_Value_Type* = "Целое"), вещественное число из множества вещественных чисел (*Attribute_Value_Type* = "Вещественное") или элемент множества {Истина, Ложь} (*Attribute_Value_Type* = "Логическое").

Axiom_Concept – понятие, представляющее аксиому в описании графовой структуры информации, оно единственно, и через него не может быть выражено ни одно другое понятие, т.е. в него не входит ни одна дуга. *Axiom_Concept* ∈ *Concepts*.

Id_Concept – понятие, представляющее идентификатор в описании графовой структуры информации, оно единственно и не может быть выражено через другие понятия, т.е. является терминальным понятием. *Id_Concept* ∈ *Concepts*.

Const_Concept – понятие, представляющее константу в описании графовой структуры информации, оно единственно и не может быть выражено через другие понятия, т.е. является терминальным понятием. *Const_Concept* ∈ *Concepts*.

Syntax_Restrictions = (*Lexica*, *Syntax*) – описание синтаксических ограничений. Оно может отсутствовать, если не требуется иметь дело с текстовым представлением информации.

$Lexica = \{<Lexem_Type_i, Definition_i>\}_{i=1}^{5}$

$Syntax = \{<Concept_i, Definition_i>\}_{i=1}^{conceptscount-2}$

$Lexem_Type_i \in$ {“Идентификатор”, “Целое число”, “Вещественное число”, “Строковая константа”, “Ограничитель строковой константы”}.

$Concept_i \in Concepts \setminus \{ Id_Concept, Const_Concept\}$

$Definition_i$ – строка символов, включающая метасимволы и представляющая конкретное лексическое или синтаксическое ограничение для конкретного понятия или вида лексемы.

Приведенная выше модель синтаксических ограничений имеет свое представление, в терминах которого пользователь может формально описывать язык текстового представления информации.

Синтаксические ограничения языка кроме, собственно, синтаксических определений содержат еще и определения лексики. Для лексики и синтаксиса может быть определено единственное ограничение для каждого элемента.

Синтаксическое определение представляет собой определение объемлющего понятия графовой структуры через объемлемые понятия и элементы конкретного синтаксиса языка текстового представления.

Лексический словарь языка определяется фиксированным набором вершин, соответствующих базовым типам данных – целое, вещественное и строковое и, кроме этого, определением вида идентификатора, или имени.

Терминальная вершина «определение», описывающая сорт *строковое*, которой соответствует лексическое или синтаксическое ограничение для понятия, представленного родительской вершиной. В сети понятий, описывающей синтаксические ограничения для конкретного языка текстового представления вершиной, соответствующей данной вершине будет являться терминальная вершина, значение которой задает конкретное лексическое или синтаксическое ограничение для конкретного понятия языка или вида лексемы.

Все ограничения на вид лексем КС-языка задаются с помощью регулярных выражений с использованием нотации perl5 [5]. Строка, представляющая синтаксическое определение, содержит специальные метасимволы, полное описание которых приводится в диссертационной работе.

Модель описания структурных проекций *MDM* представляет собой порождающую модель, определяющую семейство исчислений. Любая порождающая модель [6], [7] состоит из языка формального задания исчислений этой модели (языка для записи правил исчислений) и универсального рецепта этой модели. Последний определяется структурой состояний порождающего процесса, способом формирования начального состояния, способом построения порождающего процесса по правилам исчисления и начальному состоянию, а также правилом остановки порождающего процесса.

Любое исчисление, заданное на языке порождающей модели, определяет множество порождающих процессов генерации графовых структур. Таким образом, множество порождающих процессов, получаемых в результате выполнения универсального рецепта порождающей модели для каждого конкретного исчисления, рассматриваются в качестве модели всех возможных процессов генерации графовых структур.

Определим язык формального задания правил исчисления порождающей модели процессов генерации графовых структур как семерку

(*Concepts*, *Relations*, *Attributes*, *Computable*, *Storable*, *Loadable*, $\{\%\$\}_{i=0}^{n\geq 0}$), где

Concepts – множество понятий,
Relations – множество отношений,
Attributes – множество атрибутов,
Computable = {Истина, Ложь},
Storable = <*boolean_value*, *alias_name*>,
Loadable = <*boolean_value*, *alias_name*>,
boolean_value ∈ {Истина, Ложь},
alias_name – строка символов, представляющая псевдоним для значения атрибута, под которым нужно сохранить это значение в хеш-таблицу или по которому нужно получить это значение из хеш-таблицы.

Каждое исчисление порождающей модели задается конечным непустым множеством записанных на этом языке порождающих правил перевода исходной графовой структуры в целевую: $\pi = \{Rule_i\}_{i=1}^{rulescount}$. Каждое правило $Rule_i$ имеет вид $\alpha \to \beta$, где

$\alpha \in Concepts \cup Relations \cup Attributes$,
$\beta \subseteq Concepts \cup Relations \cup (Attributes \times Computable \times Storable \times Loadable)$.
При этом $Concepts \cup Relations \cup Attributes \neq \varnothing$ и для любых $\alpha_1, \alpha_2 \in Concepts \cup Relations \cup Attributes$, если правила $\alpha_1 \to \beta_1$, $\alpha_2 \to \beta_2$ входят в π, то $\alpha_1 \neq \alpha_2$.

Для формирования правил языка порождающей модели разработан язык описания структурных проекций, на котором пользователь может формально описывать проекции графовых структур. Каждое соответствие определяет набор понятий, отношений и атрибутов целевой графовой структуры, сопоставленный элементу исходной графовой структуры.

Специфицирован абстрактный синтаксис языка описания структурных проекций и описана его операционная семантика.

Исследование средства преобразования графовых структур при решении практических задач

Предложенный подход реализован в прототипе, предназначенном для перевода программ с одного процедурного языка программирования на другой. Прототип поддерживает языки Pascal, C, язык модели структурных программ [8], [9] и разработан в рамках системы преобразований программ в специализированном банке знаний о преобразованиях программ. Прототип реализован в среде программирования Java в среде разработки приложений IntelliJ Idea, с использованием ресурсов многоцелевого банка знаний (http://mpkbank2.dvo.ru/mpkbank/).

Основной целью экспериментов, проводимых с программным средством, является установление того, насколько возможным и целесообразным является его использование при решении задач преобразования информации, представленной графовыми структурами.

В качестве примера рассмотрим использование программного средства при решении задач перевода компьютерных программ с одного языка представления на другой.

В качестве материала для эксперимента были выбраны:

1. Графовая структура, описывающая устройство ограниченного подмножества языка программирования Pascal.

2. Графовая структура, описывающая устройство ограниченного подмножества языка программирования С.

3. Графовая структура, описывающая устройство базового языка модели структурных программ.

В табл. 1 приведены некоторые количественные характеристики графовых структур, представляющих описания этих языков.

Таблица 1 – Количественные характеристики графовых структур, представляющих описания этих языков

Характеристика / Вид информации	Количество понятий	Количество отношений	Количество атрибутов	Количество синтаксических ограничений
1	105	170		103
2	141	216		139
3	16	15	25	

В табл. 2 сведены данные о количестве соответствий в описании проекции i-го вида информации на j-й.

Таблица 2 – Количество соответствий в описании проекции i-го вида информации на j-й

Вид информации	**1**	**2**	**3**
1			58
2			79
3			

В табл. 3 приведены некоторые количественные характеристики графовых структур, представляющих программы на этих языках, а также приведено среднее время (в секундах), затраченное на выполнение анализа, генерации и синтеза соответственно.

M_s (M_t) $\in$ {1, 2, 3}.

C_s (C_t) – количество понятий в графовой структуре, представляющей программу на исходном (целевом) языке. R_s (R_t) – количество отношений в графовой структуре, представляющей программу на исходном (целевом) языке. A_s (A_t) – количество атрибутов в графовой структуре, представляющей программу на исходном (целевом) языке.

T_a – среднее время (в секундах), затраченное на анализ текстового представления программы и формирования ее структурного представления.

T_g – среднее время (в секундах), затраченное на генерацию структурного представления программы на целевом языке, на основе структурного представления этой программы на исходном языке.

T_s – среднее время (в секундах), затраченное на синтез текстового представления программы на основе ее структурного представления.

Таблица 3 – Количественные характеристики процесса преобразования графовых структур, представляющих программы на описанных языках

M_s	C_s	R_s	A_s	M_t	C_t	R_t	A_t	T_a	T_g	T_s
1	298	148		3	42	22	63	0.09	0.39	
1	528	263		3	61	20	87	0.125	0.36	
2	274	136		3	38	20	74	0.11	0.35	
2	288	143		3	51	28	78	0.114	0.353	

Замеры времени проводились при следующих условиях.

Аппаратное и программное обеспечение, используемое при проведении экспериментов:

На стороне клиента: Intel Pentium4 3 GHz, 2 Гб ОЗУ. Программное обеспечение: ОС MS Windows XP SP3.

На стороне сервера: Intel Pentium4 3 GHz, 4 Гб ОЗУ. Программное обеспечение: ОС MS Windows 2003 Server, СУБД MySQL.

Пропускная способность сетевого соединения между сервером и клиентом: 100 Мб/с.

Заключение

В работе изложен подход к преобразованию информации, представленной графовыми структурами. Представлена концептуальная схема преобразования информации, представленной графовыми структурами. На основе видов используемой информации в схеме выделено три уровня. Третий уровень объединяет модели, в соответствии с которыми формируется информация второго уровня. Второй уровень объединяет описание графовых структур и описание структурных проекций. Описание графовых структур является метаинформацией по отношению к информации первого уровня, объединяющего, в свою очередь, информацию, которая подвергается собственно преобразованию. Приведены модели, в соответствии с которыми описываются графовые структуры и проекции, которые являются спецификацией на преобразование графов.

Литература

1. Hainaut Jean-Luc. Transformation-Based Database Engineering, Transformation of knowledge, information and data: Theory and Applications / Hainaut Jean-Luc : [Patrick van Bommel]. – Information Science Publishing, 2005. – P. 1-28.
2. Partsch H. Program transformation systems / H. Partsch, R. Steinbrüggen // Computing Surveys. – 1983. – 15(3).
3. Штейнберг Б.Я. Открытая распараллеливающая система // Математические методы распараллеливания рекуррентных циклов для суперкомпьютеров с параллельной памятью / Б.Я. Штейнберг. – Ростов н/Д. : Изд-во Рост. ун-та, 2004. – С. 166-182.
4. van Deursen A. Domain-specific languages: an annotated bibliography / A. van Deursen, P. Klint, J. Visser // SIGPLAN Not. – 2000. – Vol. 35. – P. 26-36.
5. Регулярные выражения в Perl5 [Электронный ресурс]. – Режим доступа : http://www.perl.com/doc/FMTEYEWTK/regexps.html.
6. Успенский В.А. Теория алгоритмов: основные открытия и приложения / В.А. Успенский, А.Л. Семенов. – М. : Наука ; гл. ред. физ.-мат. лит., 1987. – 288 с. – (Библиотечка программиста).

7. Клещев А.С. Семантические порождающие модели. Общая точка зрения на фреймы и продукции в экспертных системах / Клещев А.С. – Владивосток : ИАПУ ДВНЦ АН СССР, 1986. – 39 с. – (Препринт).
8. Артемьева И.Л. Модель онтологии предметной области «Оптимизация последовательных программ». Ч. 1. Термины для описания объекта оптимизации / И.Л. Артемьева, М.А. Князева, О.А. Купневич // НТИ. Сер. 2. – 2002. – № 12. – С. 23-28.
9. Артемьева И.Л. Модель онтологии предметной области «Оптимизация последовательных программ». Ч. 2. Термины для описания процесса оптимизации / И.Л. Артемьева, М.А. Князева, О.А. Купневич // НТИ. Сер. 2. – 2003. – № 1. – С. 22-29.

М.О. Князєва, В.А. Тимченко

Перетворення графових структур представлення інформації

У роботі викладено підхід до перетворення інформації, представленої графовими структурами. Представлено концептуальну схему перетворення інформації на основі проекцій графових структур. Наведені моделі, відповідно до яких описуються графові структури і проекції, які є специфікацією на перетворення графів. Запропонований підхід реалізовано в прототипі, призначеному для перекладу програм з однієї процедурної мови програмування на іншу. Прототип підтримує мови Pascal, C, мову моделей структурних програм і розроблений в рамках системи перетворень програм. Прототип реалізовано в середовищі програмування Java.

M.A. Knyazeva, V.A. Timchenko

Transformation of Information Represented as Graph Structures

The paper develops and illustrates an approach to transformation of information represented as graph structures. The conceptual scheme of information transformation based on mapping of graph structures is presented. There are three levels in the scheme that allows reaching flexibility on operation with the information of different subject domains. To each level there correspond the models intended for the description of the information and the specification of ways of its transformation. The transformer can process both with structural and with textual information representations. It offers models describing graph structures and mappings which are specifications for graph transformation. This approach is implemented in the prototype that ensures program translation from one procedural programming language into another. The prototype supports such languages as Pascal, C, language of structure program models and is developed within the framework of the program transformation system. The prototype is implemented in the Java programming environment.

Статья поступила в редакцию 26.06.2009.